DECSYSTEM

BATCH
Reference Manual

Order No. DEC-20-OBRMA-A-D

20

digital

BATCH

Reference Manual

Order No. DEC-20-OBRMA-A-D

digital equipment corporation · maynard, massachusetts

First Printing, February 1976

The postage prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DIGITAL	DECsystem-10	MASSBUS
DEC	DECtape	OMNIBUS
PDP	DIBOL	OS/8
DECUS	EDUSYSTEM	PHA
UNIBUS	FLIP CHIP	RSTS
COMPUTER LABS	FOCAL	RSX
COMTEX	INDAC	TYPESET-8
DDT	LAB-8	TYPESET-10
DECCOMM	DECsystem-20	TYPESET-11

CONTENTS

CONTENTS (Cont.)

CONTENTS (Cont.)

Page

FIGURES

TABLES

PREFACE

The *DECsystem-20 Batch Reference Manual* is a complete reference document describing the commands for controlling jobs under the GALAXY Batch System. The reader of this manual is expected to be familiar with the monitor commands available to timesharing jobs and familiar with the commands to the various system programs that he will use. This reference material is not included in the *Batch Reference* manual. Therefore, it is recommended that this manual be used in conjunction with the *DECsystem-20 User's Guide.* References to all relevant material are given in the REFERENCES section at the end of this PREFACE.

The information in this manual reflects the software of release 1A of the GALAXY Batch system. This release includes:

BATCON, version 101
LPTSPL, version 101
QUASAR, version 1A
SPRINT, version 2B
DECsystem-20 Operating System, release 1

SYNOPSIS OF THE DECSYSTEM-20 BATCH REFERENCE MANUAL

Chapter 1 contains a brief introduction to the Batch system and its various components. Chapter 2 describes the several methods for submitting jobs to the Batch system. Chapter 3 gives a detailed description of each of the SPRINT commands (control cards) and describes the output produced by SPRINT. Chapter 4 contains a discussion of the Batch Controller (BATCON); its method of interpreting user input, error processing, BATCON output, and a description of each of the commands. Chapter 5 lists the Batch System error messages as well as a brief description of the meaning of each message and the suggested corrective action.

CONVENTIONS USED IN THE DECSYSTEM-20 BATCH REFERENCE MANUAL

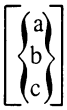
The following is a list of the symbols and conventions used in this manual:

a	An alphabetic character.
n	A decimal number.
o	An octal number.
x	An alphanumeric character.

A carriage return.

{ } Symbol indicating that one of the items enclosed within the braces must be selected.

[ ] Symbol indicating an optional element which the user may or may not include.



Therefore, the effect of a list enclosed within braces which are enclosed within brackets is that it is acceptable to have any one or none of the elements.

REFERENCES

This manual describes only the commands for controlling jobs in the Batch system. Those users who wish to have more information about system commands can refer to the *DECsystem-20 User's Guide* (DEC-20-OUGAA-A-D) referred to within this document simply as the *User's Guide*.

Error messages from system programs supplied by DEC that are invoked by the user's job are explained in the appropriate manuals. For example, if a FORTRAN program fails to compile successfully, the descriptions of the error messages the user receives from the FORTRAN compiler can be found in the *DECsystem-20 FORTRAN Reference Manual* (DEC-20-LFRMA-A-D).

Other documents referenced in this manual or which might prove useful to the Batch system user are:

DECsystem-20 MACRO Reference Manual (DEC-20-LMRMA-A-D)
DECsystem-20 COBOL Programmer's Reference Manual (DEC-20-LCRMA-A-D)
DECsystem-20 BASIC User's Guide (DEC-20-LBMAA-A-D)
DECsystem-20 APL Reference Manual (DEC-20-LARMA-A-D)
DECsystem-20 EDIT User's Guide (DEC-20-UEUMA-A-D)
DECsystem-20 Batch Operator's Guide (DEC-20-OBOGA-A-D)
DECsystem-20 Monitor Calls Reference Manual (DEC-20-OMRMA-A-D)

CHAPTER 1
INTRODUCTION

The Batch System, operating under the control of the DECsystem-20 Operating System, increases system throughput by processing jobs that do not require human interaction. Types of jobs best suited for a batch environment are: jobs that are large and long-running, jobs that require large amounts of data, jobs that are frequently run for production, and jobs that require little or no interaction with the user. Up to 127 Batch jobs can be processed concurrently by the Batch System. Batch jobs may be entered from card readers and interactive terminals.

1.1 BATCH COMPONENTS

The Batch System consists of a group of programs. Some of these programs are used for Batch operations only; others are available for various operations of the total computing system.

The individual Batch components are: the input spooler, SPRINT; the Queue Manager, QUASAR; the Batch Controller, BATCON; and the line-printer spooler, LPTSPL.

1.1.1 The Input Spooler

The Spooling Processor for Input (SPRINT)

1. reads a sequential input stream from a card reader,
2. separates the input by placing it in files according to the SPRINT control cards contained in the input stream,
3. creates the job's log file and enters a report of its processing,
4. spools card decks to disk with no Batch job created, and
5. enters the job into the Batch input queue (if the job requires it) for subsequent processing by the Batch controller.

SPRINT provides the facility for reading ASCII and image cards. (Refer to Appendix C for a table of card codes.)

1.1.2 The Queue Manager

QUASAR, the Batch and spooling system queue manager, is the program that builds and maintains the system queues of tasks to be processed by the Batch Controller and the output spooler. QUASAR communicates with other components in the Batch system through the Inter Process Communication Facility (IPCF). QUENCH (when it has been requested to queue a job) or SPRINT (when it completes reading in a job) informs QUASAR, which makes an entry into the Batch input queue. QUASAR schedules the jobs in the Batch input queue to be run by BATCON.

Scheduling consists of computing and dynamically revising priorities for the job, according to the job's parameters and the priorities established by the system. While the job is running, its queue entry is flagged to show it is in use, but the entry is not deleted from the queue until the job terminates. When the job is logged off the system, an output queue entry is usually made; and the entry in the input queue is deleted. QUASAR schedules jobs in the output queues to be processed by LPTSPL. The job's output queue entry is deleted only when the output is completely finished.

1.1.3 The Batch Controller

The Batch Controller, BATCON, controls all jobs entered into the Batch System. It reads the control file created either by SPRINT or by the user and initiates and controls the running of the job by passing data and system program commands directly to the job.

The Batch Controller makes entries to the log file to record its processing of the control file and the job.

1.1.4 The Line-Printer Spooler

The output spooler, LPTSPL, is the program that drives the line printer. This program processes a queue (or list) of requests which is maintained by QUASAR. Instead of sending output directly to the line printer, the user or program sends output to the disk and the output spooler later transfers the information to the line printer.

One advantage of this procedure is that if the output device is unavailable (for example, because another program is using it), the user need not know; his program simply sends its output to the disk. Another advantage is that since the disk is a high speed device, the user's program is not delayed while it is writing on a slow speed device.

There are two methods of entering jobs into a queue for processing by LPTSPL.

1. Spooling output to a device – this is simply writing to a spooled line printer. For example,

   @LIST MYPROG.FOR

   (this is equivalent to the command @COPY (FROM) MYPROG.FOR (TO) LPT:)

2. Explicitly queuing output to a device – this method involves using the queue-class command PRINT, which runs the QUENCH program. For example,

   @PRINT MYPROG.FOR

CHAPTER 2
SUBMITTING JOBS

A job is a unit that consists of one step or a group of steps that the user initiates. It can contain

1. a single program and its related data, or several programs and their data, and/or
2. the system and program-level commands that are required to control the programs or supply information to the user.

The Batch System allows the user to submit his job by one of the following two methods:

1. The user punches his job on cards, inserts control cards to SPRINT, and leaves his cards at the designated place for the operator to run (refer to paragraph 2.1).
2. The user bypasses SPRINT by creating his own control file on disk for the Batch Controller and then enters his job into the Batch input queue from his terminal (refer to Paragraph 2.2).

2.1 SUBMITTING A JOB WITH CARDS

With this method a job is submitted via a deck of cards, bounded by the control cards that mark its beginning and end. Other control cards to SPRINT are interspersed among the card deck to direct SPRINT's processing. Figure 2-1 shows a job containing the appropriate control cards to SPRINT. This job compiles, loads, executes, and lists a FORTRAN program.

The SPRINT control cards shown in Figure 2-1 are just a few of the control cards available to the user. For a complete description of all the SPRINT control cards, refer to Chapter 3.

2.1.1 The $JOB Card

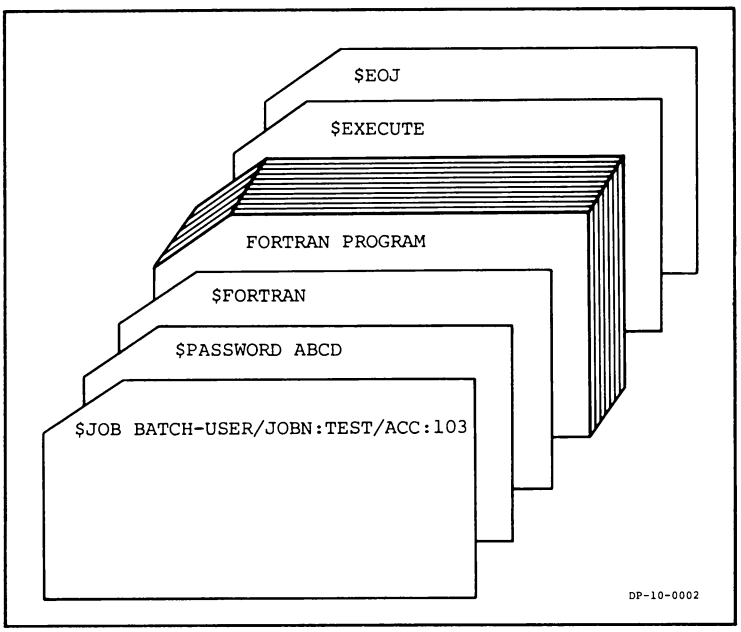
This card notifies SPRINT that a job is to be processed. SPRINT creates a control file into which commands are placed for the Batch Controller and it creates a log file on the disk. The first argument (BATCH-USER) shown on this card is the user name. The first switch (/JOBNAME:TEST) establishes the user-assigned name for the job. TEST will be the name used for both the control and log files with file types of .CTL and .LOG respectively. The /ACCOUNT switch and the account number are required. If the switch is omitted, an error message is issued and the job is not processed. For a description of switches which can be used on this card, refer to Chapter 3.

2.1.2 The $PASSWORD Card

This card contains the PASSWORD associated with the user name and account number specified on the $JOB card. In Figure 2-1, the PASSWORD is ABCD, which was assigned to the user by the system manager. Refer to Chapter 3 for more information on the $PASSWORD card.

2.1.3 The $FORTRAN Card

This card causes SPRINT to insert a COMPILE monitor command (refer to the *User's Guide*) into the control file to cause the program to be compiled. Immediately following the $FORTRAN card is the FORTRAN source program to be compiled. The source program is read into a disk file with a unique filename created by SPRINT and with a file type of .FOR to indicate that the source program is a FORTRAN program. Refer to Chapter 3 for more information on the $FORTRAN card.



This figure illustrates, and Paragraph 2.1.3 describes, a job which happens to contain a FORTRAN program. With the substitution of the appropriate *$-language* card for the $FORTRAN card, this arrangement would be equally valid for other supported languages.

Figure 2-1 Typical Job on Cards

2.1.4 The $EXECUTE Card

The card after the FORTRAN program is the $EXECUTE card. This card causes SPRINT to insert an EXECUTE system command (refer to the *User's Guide*) into the control file in order to load and then execute the previously compiled program. Refer to Chapter 3 for additional information on this card.

2.1.5 The $EOJ Card

The last card shown in the example is the $EOJ card. It signals the end of the job. Refer to Chapter 3 for more information about this card.

2.1.6 Output

Once the program is punched on cards, the card deck is submitted to the operator, who in turn stacks the job in the card reader. The user receives his output in the form of line printer listings. Refer to Chapter 3, section 3.2 for an explanation of the job output.

2.2 SUBMITTING A JOB (CONTROL FILE) TO THE BATCH CONTROLLER

A job (i.e., a control file) may be submitted directly to the Batch Controller. The file must be a disk file and must be created with the system editor, EDIT. The control file consists of system commands, Batch commands, user program commands, comments, and sequence control statements. Refer to Chapter 4, section 4.4 for a description of Batch control file commands. The following is an example of creating a control file. It assumes that a file named DATA.FOR already exists on disk.

```
@CREATE (FILE) JOB.CTL↵
Input: JOB.CTL.1
00100    @EXECUTE /COMPILE PROG.FOR /LIST$
*E↵

[JOB.CTL.1]
```

(Note that the dollar sign at the end of line 00100 is what is echoed by the system when the user types ESCape.)

The control file here is named JOB.CTL. Once the control file is created, the user enters the job into the Batch input queue with the SUBMIT system command. (Refer to the *User's Guide* for a more detailed description of the SUBMIT command.) The general form of the SUBMIT command is

```
SUBMIT jobname = control file, logfile
```

The control file created in the previous example could be submitted to the Batch system using the following abbreviated form of the SUBMIT command.

```
@SUBM JOB.CTL
```

2.3 SUBMITTING RELATED JOBS

Jobs are not necessarily run in the order that they are read into the Batch System. Priorities stipulated by the user on the $JOB card (refer to Chapter 3) and additional parameters set by the Batch System are dynamically computed by QUASAR to determine in what order the jobs are run. However, it is often useful to submit several jobs that must be run in a specific order; for example, one job updates a master file and another job processes it. Therefore, the running of one job is dependent upon the running of the other. Although these jobs could be combined into one large job, it is sometimes more convenient to keep them distinct; i.e., they might be submitted by different people at different times.

Because the jobs in the Batch System are run in order of priority, the user specifies an additional priority called an initial dependency count on the $JOB card of the dependent job. This dependency count becomes part of the queue entry. Any input queue entry that has a dependency count greater than zero cannot be scheduled. When the count becomes zero, the job becomes eligible to be run by the Batch Controller. The dependency count can be altered by including the QUEUE command as part of any job upon which the dependent job is waiting. Refer to the SUBMIT command in the *User's Guide* for more information.

The QUEUE command switch that allows the user to change the dependency count of another job is the /MODIFY/DEPEND:*n* switch. If the user specifies a plus or minus sign before the count (*n*), that number is added to or subtracted from the dependent job's count. If the subtraction results in a negative number, then it is treated as if it were zero. If the user does not specify a sign, the dependent job's count is changed to the count specified in the /MODIFY/DEPEND:*n* switch.

CHAPTER 3
SPRINT—THE INPUT SPOOLER

SPRINT reads a sequential input stream and separates the input into appropriate files, according to the control cards supplied by the user. SPRINT prepares the user's input for processing, but it does not initiate any processing itself. This processing is accomplished by the Batch Controller.

The input to SPRINT can include user programs and data, system commands, Batch Controller commands and SPRINT control cards.

SPRINT creates three types of files during its copying of the input data: the user's data files and source programs, the Batch control file, and the job's log file. SPRINT creates the data files and source programs according to the control cards in the input and places them into the user's directory. Programs and data are acted upon by the contents of the control file and are passed to the running job by the Batch Controller. Refer to section 3.1 for the description of the control cards that cause SPRINT to copy information into these files.

A user control file is created for each valid job and is subsequently processed by the Batch Controller. This file contains all system level and user level commands encountered in the input. SPRINT enters commands resulting from the processing of its control cards and the control file is placed in the user's directory. Refer to Chapter 4, section 4.4 for a description of the Batch Controller commands that can be entered into the control file.

The job's log file contains a report of SPRINT's processing, along with a record of any operator intervention during its operation. This file is in the user's disk area along with the other SPRINT-created files and usually is deleted after it is printed by the line-printer spooler.

3.1 SPRINT CONTROL CARDS

Control cards are interspersed through the input stream to direct SPRINT in separating the input into the appropriate files, either the user's data files or the control file processed by the Batch Controller. The control cards contain a dollar sign ($) in column 1 and an alphabetic character in column 2. These are the only cards read and interpreted by SPRINT, the remainder of the input is separated and placed into the appropriate file. Of these control cards, the $DATA, $CREATE, and $-*language* without a filename specified cards are also referred to collectively as File Cards, in that they all cause subsequent cards up to the next SPRINT control card (except $MODE) to be placed into a disk file. Note that if the user creates his own control file, he bypasses SPRINT and, therefore, cannot use these control cards.

Only the first few letters of the card name or switch need be specified; as long as the name is unique within its class, it is accpeted. The first three characters of a card name are generally sufficient to ensure uniqueness. However, it is recommended that card and switch names be spelled out completely in Batch jobs since if any of the programs change (e.g., add a new card or command), existing card decks may be come invalid. The standard comment and continuation conventions for the system can be used on the control card. The exclamation point (!) indicates the beginning of a comment and all the characters following the exclamation point through the end of card are treated as a comment. While a comment field cannot actually be continued, the same effect can be accomplished by the use of an additional card that contains only a comment. Some examples follow:

```
$CREATE MYFILE.FOR                    !BEGINNING OF SOURCE FILE
            .
            .                         } FORTRAN source deck
            .
$EOD                                  !END OF SOURCE FILE
$FORTRAN MYFILE.FOR
$!BEGIN COMPILATION
```

The user may continue operative information by placing a hyphen as the last non-TAB, non-space character before the end-of-card or before the beginning of a comment (if present). That is, the hyphen will indicate continuation on the next card if it immediately precedes one of the following:

1. the end of the card, or
2. a string of spaces and/or TABs followed by the end of card, or
3. an exclamation point followed by a comment, or
4. a string of spaces and/or TABs followed by a comment.

The following two examples illustrate the continuation of operative information, first without comments and then with comments, respectively:

1. $CREATE MYFIL1.DAT/ASCII–
   /SUPPRESS

2. $CREATE MYFIL2.DAT–!THIS IS A COMMENT
   /ASCII/SUPPRESS

All defaults for control card parameters are installation parameters.

The following pages describe the SPRINT control cards. The cards are listed in alphabetical order with the exception of the $-*language* cards (e.g., $COBOL) which are all grouped together under the heading, $-*language*.

$COBOL

See the $-*language* card for information about the $COBOL card.

$CREATE

Function
The $CREATE card directs the Input Spooler to copy all cards following it into a data file. Copying is terminated by different cards depending on whether the /DOLLARS or the /NODOLLARS switch is specified. Refer to the brief switch descriptions below, or to Appendix A for a more detailed explanation. The $CREATE card also has the capability of placing the specified file into the line-printer output queue.

The $CREATE card followed immediately by a $EOD or any other appropriate terminator will create a null file on disk.

Card Format
$CREATE *dev:name.typ*/$sw_1$/$sw_2$. . . /$sw_n$

dev: = a device name which may be either DSK: or a system logical name. The default is DSK:.

name.typ = the user-assigned name and type of the file to be created. If omitted, the name CR*xxx* (where *x* represents a character arbitrarily chosen by SPRINT to produce a unique name) is created by SPRINT.

The following is a brief description of each of the switches which can be used on the $CREATE card. These switches are explained in detail in Appendix A.

Switch	Meaning
/ASCII	The input is read in ASCII card code.
/DOLLARS	Only the $SEQUENCE, $JOB, $EOD, and $EOJ control cards are recognized. All other cards with a dollar sign in column 1 are treated as user data.
/NODOLLARS	If a card has a dollar sign in column 1, the contents of column 2 determine whether it is treated as a SPRINT control card or user data.
/IMAGE:*nn*	The card deck is read in image mode. The switch must be followed by a decimal number in the range 2 through 80.
/PRINT	SPRINT places the disk file created by the $CREATE card into the line printer output queue.
/SUPPRESS	Trailing blanks are suppressed.
/NOSUPPRESS	Trailing blanks are not suppressed.
/WIDTH:*nn*	Columns 1 through *nn* (inclusive) of each card are read.

$DATA

Function
This card is a File Card which precedes a data deck and causes SPRINT to copy the data into a file on the user's disk area and to insert an EXECUTE system command into the control file.

SPRINT maintains a list of filenames of all source or relocatable programs that have been processed since the beginning of the job or since the last $DATA or $EXECUTE card was read. Each time a program is copied by SPRINT, its name is placed in the list. When the $DATA card is read, SPRINT places an EXECUTE command into the control file and copies the filenames of the programs into the EXECUTE command string. When the next $-*language* or $INCLUDE card is encountered, SPRINT clears the list of filenames so that the next entries into the list reflect only those filenames copied since the last $DATA or $EXECUTE command was read.

When the job is run, the programs are loaded and executed. No compilation is performed because the programs are either in relocatable binary form or have been previously compiled because of the $-*language* card. If two $DATA cards appear in a row, the same programs are reloaded and executed again.

Card Format
$DATA *dev:name.typ*/$sw_1$/$sw_2$. . . /$sw_n$

dev: = a device name which can be either DSK: or a system logical name. If omitted, DSK: is assumed.

name.typ = the name of the file to be created. If omitted, SPRINT creates the name *xxx* (where *x* represents a character arbitrarily chosen by SPRINT to produce a unique filename) with the type .CDR and creates a spooled card-reader file. That is, the user may read the data file simply by referencing the card-reader from his program. However, when a name is specified, the user must specifically read the data file from disk.

/sw_1/sw_2. . . /sw_n = switches that control the mode of reading and interpreting of the input stream.

The following is a brief description of the switches which can be used on the $DATA card. These switches are explained in detail in Appendix A.

Switch	Meaning
/ASCII	The input is read in ASCII card code.
/DOLLARS	Only the $SEQUENCE, $JOB, $EOD, and $EOJ control cards are recognized. All other cards with a dollar sign in column 1 are treated as user data.
/NODOLLARS	If a card has a dollar sign in column 1, the contents of column 2 determine whether it is treated as a SPRINT control card or user data.
/IMAGE:*nn*	The card deck is read in image mode. The switch must be followed by a decimal number in the range 2 through 80.
/MAP	This switch causes a loader map to be generated and printed.
/NOMAP	A loader map will not be generated.
/SUPPRESS	Trailing blanks are suppressed.
/NOSUPPRESS	Trailing blanks are not suppressed.
/WIDTH:*nn*	Columns 1 through *nn* (inclusive) of each card are read.

The defaults for all modes are reset by the next $MODE card or by individual switches in other control cards such as in the $CREATE card and $JOB card.

Restrictions
This card can be used only when the programs in the job have been entered with a $-*language* or $INCLUDE card, since SPRINT maintains a list of the filenames of programs that are input with these commands. If the user wishes only to have the programs compiled, no $DATA or $EXECUTE card or EXECUTE command should appear in the job.

$EOD

Function
This card terminates the input that is being copied into a data file by SPRINT because of a preceding File Card. If input is not being copied and this card is read, SPRINT ignores it.

Card Format
$EOD

$EOJ

Function
This card is the last card in a job deck and it terminates the job.

Card Format
$EOJ

$ERROR
$NOERROR

Function
These cards are used to aid the Batch Controller in processing errors. They cause SPRINT to insert an @IF statement into the control file; e.g., @IF (ERROR) or @IF (NOERROR). Refer to Chapter 4, section 4.4 for an explanation of the @IF statement. These cards must appear at the points where possible errors are anticipated.

Card Formats
$ERROR *statement*

$NOERROR *statement*

statement = an executable system or Batch command preceded by an at sign (@). If the statement directs the Batch Controller to go to a statement label, the statement label line and any related lines must be included in the sequence of commands at the place the user wants the commands executed. For example,

```
$FORTRAN TEST1
$ERROR @GOTO A
$DATA TEST1.DAT
.
.
.
$ERROR @GOTO A
$TOPS20
A:: !RESUME HERE
$FORTRAN TEST2
```

$EXECUTE

Function
This card causes SPRINT to insert an EXECUTE system command into the control file. It performs the same function as the $DATA only it does not have a data deck following it. The $EXECUTE card is used when there is no data or when the data file already exists on disk (for example, through the previous use of a $CREATE card). The files to be placed in the EXECUTE command string are determined in the same way as they are for the $DATA.

Card Format
$EXECUTE [{/MAP /NOMAP}]

Switch	Meaning
/MAP	A loader map will be generated and printed.
/NOMAP	This is the complement of the /MAP switch. /NOMAP is the default.

$FORTRAN

See the $-*language* card for information about the $FORTRAN card.

$INCLUDE

Function
This card causes relocatable binary files that already exist on disk to be loaded with the user's programs. A file specified on a $INCLUDE card is added to the list of filenames remembered by SPRINT and included in the EXECUTE command string generated by a $DATA or by a $EXECUTE card.

Card Format
$INCLUDE *dev:name.typ* [/SEARCH]

dev: = the name of the device that contains the file to be loaded. This name may be either DSK: or a system logical name. If *dev:* is omitted, DSK: is assumed.

name.typ = the name of the file to be loaded. The type is normally .REL. If the filename is omitted, an error message is issued and the job is terminated.

Switch	Meaning
/SEARCH	This switch specifies that the file is to be loaded in library search mode. This switch is optional.

Restrictions
The file specified on the $INCLUDE card must be a relocatable binary file and must already exist.

$JOB

Function

This card, in conjunction with the $PASSWORD card (if required), causes SPRINT to create a control file in the user's directory into which commands are placed for the Batch Controller. In addition, SPRINT creates a LOG file in the user's directory.

SPRINT uses the name specified on the $JOB card for the control and log files with the types .CTL and .LOG respectively. If the jobname is omitted, SPRINT creates a unique name for the job. Refer to the /JOBNAME switch below for a more detailed description. In general, the jobname used on input appears in the output queues.

Card Format

$JOB *user name*/$sw_1$/$sw_2$. . . /$sw_n$

user name = The assigned user name of the person who submitted the job. This field can contain from 1-39 alphanumeric and special characters. This argument is required.

/sw_1/sw_2. . . /sw_n = switches taken from the following group. These switches are optional except for the /ACCOUNT switch.

The following is a brief description of each of the switches which can be used on the $JOB card. These switches are explained in detail in Appendix A.

Switch	Meaning
/ACCOUNT:*xxx*	This switch specifies the account to which the resources used by this job are to be billed. **NOTE** This switch is required.
/AFTER:*date time*	The job cannot be run until after the specified date and time.
/AFTER:+*hh:mm*	The job cannot be run until after the input time plus the amount of time specified.
/DEADLINE:*date time*	The job must be started by the specified date and time.
/DEADLINE:+*hh:mm*	The job must be started by the indicated amount of time after it is input.
/DEPEND:*nn*	Initial interjob dependency count (in decimal).
/ERROR:*nn*	Specifies the number of Hollerith errors permitted before the job is terminated.
/JOBNAME:*xxx*	Specifies a 1-6 character user-assigned name for the job.
/LOGDISP:*arg*	This switch specifies the disposition of the LOG file after the Batch job has been processed. The arguments are DELETE (default) and PRESERVE.
/OUTPUT:*xxx*	This switch determines whether or not the LOG file is printed. The valid arguments are LOG (default), NOLOG, and ERROR.
/PAGES:*nn*	The maximum number of pages (in decimal) that can be printed by the job.

Switch	Meaning
/PRIORITY:*nn*	The external priority of the job.
/RESTART	Specifies that the job may be restarted.
/NORESTART	Specifies that the job may not be restarted.
/SEQUENCE:*nn*	The job's sequence number within the input queue.
/TIME:*hh:mm:ss*	The limit placed on the amount of CPU time used by the job. The default is 5 minutes.
/UNIQUE:*n*	Specifies how jobs are to be protected from the effects of other jobs running in the same directory. An argument of 0 means no protection and 1 (the default) means only one Batch job at a time may be run.

The following rules apply to all the switches in the above list that require a time and/or date to be specified.

When specifying a time of day (*hh:mm:ss*)

1. The colon (:) or colons must be included.
2. Times will be right-aligned before they are interpreted. That is, if all fields are not present, the right-most field is interpreted as the number of seconds in the case where an argument of *hh:mm:ss* is required, and the right-most field is interpreted as the number of minutes when an argument of *hh:mm* is required. Some examples are given below.

/TIME:30	means	30 seconds,
/TIME:45:00	means	45 minutes and
/TIME:1:15:00	means	1 hour and 15 minutes

When specifying a date, the format is *dd-mm-yy* and:

1. The hyphen (-) must be included.
2. At least the day and the month are required.

NOTE

The month, "*mmm*", must be a 3-letter month abbreviation (e.g., JAN for January). Other date formats (such as 07 for July) are accepted but are not recommended since they could be ambiguous.

3. If the year is omitted, the date (and its associated time, if present) will be interpreted to mean the next occurrence of that date (and time).
4. If the time argument is omitted from a date specification, the time is assumed to be midnight on the specified date.

   In the examples below the current date of 6-JAN-75 will be assumed.

/AFTER:17-OCT-75	means midnight on October 17, 1975
/AFTER:17-OCT	means midnight on October 17, 1975
/DEADLINE:3-JAN 18:00	means 6 P.M. on January 3, 1976

Requirements

The $JOB card must immediately follow the $SEQUENCE card or be the first card if the $SEQUENCE card is not required.

$-language

Function

The $-*language* cards direct the Input Spooler to compile the user's programs using the appropriate language processor or interpreter. The $-*language* cards and the types they use are:

Card Name	Default File Type
$COBOL	.CBL
$FORTRAN	.FOR
$MACRO	.MAC

The following description applies to all the $-*language* cards; the user need only substitute the name of the language desired (from the above list) immediately after the dollar sign.

The $-*language* card can be used with two different input conditions.

1. The source code immediately follows the $-*language* card and the $-*language* card does not have a filename specified. In this case, the $-*language* card directs the Input Spooler to copy the program onto disk, assign the program a unique filename of the form LN*xxxx* with the type shown in the above table, and insert a COMPILE system command into the control file. The file being copied onto disk is terminated by different cards depending on whether the /DOLLARS or /NO-DOLLARS switch is specified. Refer to the brief switch descriptions below, or to Appendix A for a more detailed explanation. The source and object files will be explicitly deleted at LOGOUT time.

2. The source code of the program is already on disk and the user has specified a filename on the $-*language* card. In this case, the $-*language* card directs the Input Spooler to insert a COMPILE system command into the control file.

NOTE

A $-*language* card with a filename specified, followed by a deck containing the source code, will cause an error message to be issued.

The $-*language* card does not cause execution of the program. Execution is initiated by an EXECUTE system command or by a $DATA or a $EXECUTE card.

Card Format

If the source deck follows the $-*language* card, the format is:

$-*language(processor switches)*/sw_1/sw_2. . . /sw_n

If the source code already exists on disk and if only compilation is desired, the format is:

$-*language dev:name.typ(processor switches)* [{/LIST /NOLIST}]

dev: = the name of the device that contains the program to be compiled if the program already exists. Otherwise, this is the name of the device onto which the program is copied. If *dev:* is omitted, DSK: is assumed.

name.typ = the name of the file to be compiled. The name is specified only when the file already exists. If a filename is specified and there is a card deck to follow the $-*language* card, an error message will be used.

(processor switches) = the switches to be passed to the compiler. (Refer to appropriate language manual and to the COMPILE command for a description of the processor switches.) They must be enclosed in parentheses and must not be preceded by a slash.

$/sw_1/sw_2 \ldots /sw_n$ = the switches that control the mode of input interpretation and the listing of the compiled program. The only switch available when the program is already on disk is either the /LIST or the /NOLIST switch.

The following is a brief description of the switches that can be used with the $-*language* cards. These switches are explained in detail in Appendix A. Each of the $-*language* cards can contain any of the following switches, except in the case of the /CREF switch which can be used only with the $FORTRAN and $MACRO cards.

Switch	Meaning
/ASCII	The input is read in ASCII card code.
/CREF	A cross-referenced listing is created to be processed by the CREF program. This switch is available only on the $FORTRAN and $MACRO cards.
/DOLLARS	Only the $SEQUENCE, $JOB, $EOD, and $EOJ control cards are recognized. All other cards with a dollar sign in column 1 are treated as user data.
/NODOLLARS	If a card has a dollar sign in column 1, the contents of column 2 determine whether it is treated as a SPRINT control card or as user data.
/LIST	A compilation listing will be generated. /LIST is the default.
/NOLIST	No listing file of the program will be created.
/SUPPRESS	Trailing blanks are suppressed.
/NOSUPPRESS	Trailing blanks are not suppressed. /NOSUPPRESS is the default.
/WIDTH:*nn*	Columns 1 through *nn* (inclusive) of each card are read.

$MACRO

See the $-*language* card for information about the $MACRO card.

$MESSAGE

Function
The $MESSAGE card causes the supplied text to be output to the system operator's terminal at the time the job is run.

Card Format
$MESSAGE/$sw_1$ *text*

sw_1 = switch to indicate whether operator response is required or not. The switch is optional but, if present, must immediately follow the command ($MESSAGE) with no intervening blanks. Otherwise, an error message will be issued and the job will be terminated.

text = the message to be output to the system operator.

Switch	Meaning
/WAIT	The job will wait for a response from the operator before resuming its processing.
/NOWAIT	The job will continue after typing the message without waiting for a response from the operator. This is the default.

$MODE

Function
This card causes SPRINT to change the mode in which it is interpreting the input stream. The $MODE card can be placed anywhere after the $PASSWORD card in the command sequence. All the switches take effect immediately after the $MODE card is read. The /ASCII switch remains in effect until the next control card is read but the effects of the /NOSUPPRESS, /SUPPRESS and /WIDTH switches are not terminated until either the end of the job or until they are changed by another $MODE card.

Card Format
$MODE /$sw_1$/$sw_2$. . . /$sw_n$

/sw_1/sw_2. . . /sw_n = switches that control the mode of reading and interpreting of the input stream.

Switch	Meaning
/ASCII	The input is read in ASCII card code.
/SUPPRESS	Trailing blanks are suppressed.
/NOSUPPRESS	Trailing blanks are not suppressed.
/WIDTH:*nn*	Columns 1 through *nn* (inclusive) of each card are read. The remaining columns are truncated.

NOTE
There are no defaults for switches on a $MODE card because if a switch is not specified, the previously established value remains unchanged.

$PASSWORD

Function
This card contains the password associated with the user name specified on the $JOB card. If the password does not match the password stored in the system for the specified user name, SPRINT does not create any files, issues an error message on the LOG file, and cancels the job. This card may not be required at all installations.

Card Format
$PASSWORD *password*

password = a password comprised of from 1-39 alphanumeric characters.

NOTE
There must be exactly one space between the end of the card name ($PASSWORD) and the first character of the actual password.

Requirements
If the $PASSWORD card is required, it must immediately follow the $JOB card.

$SEQUENCE

Function
This card specifies the job's unique sequence number. The use of this card depends on the requirements in effect at the individual installation.

Card Format
$SEQUENCE *n*

n = a decimal number.

Requirements
If the installation requires this command, it must be the first card in the input stream.

$TOPS20

Function
The $TOPS20 card directs the Input Spooler to copy all cards following it into the Batch control file. The cards following a $TOPS20 card are expected to contain Batch Controller and system commands. These cards will not be processed by SPRINT. Copying of the deck is terminated by different cards depending on whether the /DOLLARS or the /NODOLLARS switch is specified. Refer to the brief switch descriptions below, or to Appendix A, for a more detailed description.

NOTE

A single system or Batch command or a group of consecutive system and/or Batch commands must be preceded by a $TOPS20 card and followed by the appropriate terminating card. If such commands are placed into the SPRINT input stream but are not properly delimited, an error message will be issued.

Card Format

$TOPS20/$sw_1/sw_2 \ldots /sw_n$

$/sw_1/sw_2 \ldots /sw_n$ = switches that control the modes in which the input is read and interpreted.

The following is a brief description of each of the switches which can be used on the $TOPS20 card. These switches are explained in detail in Appendix A.

Switch	Meaning
/ASCII	The input is read in ASCII card code.
/DOLLARS	Only the $SEQUENCE, $JOB, $EOD, and $EOJ control cards are recognized. All other cards with a dollar sign in column 1 are treated as user data.
/NODOLLARS	If a card has a dollar sign in column 1, the contents of column 2 determine whether it is treated as a SPRINT control card or user data.
/SUPPRESS	Trailing blanks are suppressed.
/NOSUPPRESS	Trailing blanks are not suppressed.
/WIDTH:*nn*	Columns 1 through *nn* (inclusive) of each card are read.

3.2 SPRINT OUTPUT

The output from a SPRINT job is normally in the form of printed listings containing the user's job output, compilation listings, and the LOG file indicating the processing performed by the programs in the Batch system. The results from the job and the LOG file are automatically placed in the queue for the line printer spooler, LPTSPL, unless the job was submitted with the /OUTPUT switch. However, the user can output to any device in the system.

Compilation listings are produced from the $-*language* control cards unless the user specifies otherwise. These listings are automatically spooled to the line printer. The user can also include the COMPILE system command in his job with switches to produce listings.

3.2.1 The Log File

As part of its processing, SPRINT creates a LOG file for each job so that the user can examine the processing performed by the Batch system. The LOG file contains a record of SPRINT's processing, control cards, any errors detected, and any operator actions.

When the job is run, the Batch Controller places additional messages into the LOG file. (Refer to Chapter 4, Section 4.6.) These messages contain a record of each line of the control file as passed to the job, every line of output that the user job sends to its controlling terminal, Batch Controller messages, and a record of most operator actions affecting the job. Note that the LOG file is appended to for jobs of the same name; thus it may be necessary to delete this file before running another job with the same name.

3.2.2 SPRINT Messages

SPRINT places six kinds of messages into the LOG file. The first part of each message, in columns 1 through 8, contains the time of day that SPRINT places the message into the file; an identifying word is located in columns 10 through 14. The body of the message is in columns 17 through 132. The identifying word for each kind of message is taken from the following group:

STDAT – gives the date, system name, SPRINT version and the input device.

STMSG – identifies any SPRINT non-error message.

STERR – identifies any SPRINT error message.

STCRD – identifies SPRINT control cards.

STSUM – identifies the summary message at the end of the job.

STOPR – identifies any operator actions that occurred during SPRINT's processing.

The first entry in the LOG file always contains the identifier STDAT and a message giving the date, the system name, the current version of SPRINT, and the input device.

The STDAT entry is followed by two STCRD lines containing the $SEQUENCE (if present) and the $JOB cards, respectively. For security reasons, the $PASSWORD card is never printed.

Each SPRINT control card is written into the LOG on an STCRD line and, if the card is a File Card, the line immediately following contains a message of the form.

14:17:40 STMSG *file.typ* Created - *nn* Cards read - *nn* Pages written

When the $EOJ card is read, SPRINT prints a summary message indicating the status of the job (whether it was completed or aborted) and giving the number of cards read, the number of files and pages written, and the number of errors that occurred. The summary is also placed in the system accounting file. Two examples of job summaries are given below.

```
1.  14:17:42  STSUM End of Job Encountered
    14:17:42  STSUM 423 Cards Read
    14:17:42  STSUM 4 Hollerith errors
    14:17:43  STSUM Batch Input Request Created


2.  16:18:22  STSUM Job Aborted due to Fatal Error
    16:18:22  STSUM 10 Cards Read
    16:18:23  STSUM 10 Hollerith Errors
```

Between the beginning and ending messages, SPRINT prints any operator actions as they occur, some nonerror messages, and reports of errors it has detected.

3.2.3 SPRINT Error Reporting

SPRINT places messages in the LOG file which describe errors that have occurred during its processing.

The messages fall into three categories:

1. ? at the beginning of a message indicates a fatal error. No further processing will be done and the job will be terminated. For example, an unrecognizable control card would generate a fatal error message.

2. % at the beginning of a message indicates an advisory or warning message. Processing will continue in this case. For example, an unrecognizable switch, which can be ignored, would generate a warning message.

3. [at the beginning of a message indicates a comment from SPRINT to the user. It is for the user's information only and does not affect the running of the job.

In the LOG file, SPRINT error messages are preceded by the identifier STERR, followed by the message type indicator (?, %, or [). Next is a 6-character prefix of the form SPT*xxx*. SPT indicates that SPRINT issued the message and *xxx* is an abbreviation of the text of the message. The remainder of the line contains the message itself. For example,

```
STERR ?SPTTMH   TOO MANY HOLLERITH ERRORS
```

Chapter 5 contains a complete list of SPRINT messages.

CHAPTER 4
BATCON—THE BATCH CONTROLLER

The Batch Controller controls all jobs entered into the Batch System. It reads the control file and initiates and controls the running of the job.

4.1 THE CONTROL FILE

The control (CTL) file is read and processed by the Batch Controller. (Refer to Chapter 2 for an explanation of control file creation and job submittal.) The control file can contain system commands, Batch commands, data for a user or system program, or comments.

The Batch Controller passes the data and system commands to the appropriate program or to the system for action. The Batch commands are interpreted and handled by the Batch Controller itself.

4.1.1 Interpretation of Line Identifiers

BATCON determines the destination of a line in a control file by interpreting the first character encountered after each carriage return/line feed (CR/LF) in the control file. The Batch Controller interprets these "first characters" as follows:

*(asterisk)	This line is to be treated as user data or data to a system program (i.e., a user-level command). The Batch Controller suppresses the asterisk and sends the remainder of the line to the job. However, if the job is currently at system command level, the line is treated as a comment.
= (equals sign)	This is interpreted in the same manner as the asterisk. However, the equals sign indicates that the user wants to inhibit the Batch Controller from transmitting the CR/LF terminating this line. For example, in the DDT sequence =FOO/ the "/" is a terminator to DDT.
@ (at sign)	The interpretation of the at sign depends on the next non-blank, non-TAB character. If that character is numeric, the line, including the at sign, is treated as user data. If the job is at system command level, the line is treated as a comment. If that character is an alphabetic or a special character, the line is either a system command or a Batch command as defined in Section 4.4. If it is a system command, the job is placed at system command level, the at sign is suppressed, and the line is sent to the system. If it is a Batch command, it is processed by the Batch Controller. (Refer to Section 4.4.)
! (exclamation point)	The exclamation point is used to identify a comment line. Comment lines are placed in the LOG file and do not affect the running program.
% (percent sign)	This character identifies the labels reserved for use in association with the occurrence of errors. (Refer to Section 4.3.1.)

FF (form-feed)
VT (vertical tab)

If either of these characters appears as the "first character" in a line of a control file, that character is treated as user data and the character immediately following it is treated as if it were in the "first character" position (i.e., examined by the Batch Controller).

Form-feed and vertical tab do not affect the character that follows when they occur in any position other than the "first character" position in a line in a control file.

NOTE

This interpretation enables the user to enter form-feed and vertical tab into thecontrol file in order to align the LOG file output.

Lines in the control file that begin with any other special character (including blank and TAB) are treated as user data, and the initial character is considered part of that data. If the job is currently at system command level, the whole line is treated as a comment.

Any line in the control file that begins with an alphabetic character (and that is not a label – refer to Section 4.2) is interpreted as follows:

1. If the job is running a user or system program, the line is passed to the program.
2. If the job is at system command level, the line is treated as a Batch or command as defined in Section 4.4.

NOTE

Because of the two possible interpretations, it is suggested that users identify each line to clarify its meaning.

One other special character is used to denote control characters. It is the circumflex (^). It is interpreted as follows, regardless of its position in a line in a control file.

If the character following the circumflex is numeric, the circumflex and the digit are passed to the job. For example:

^2 remains^2

If the character following the circumflex is alphabetic, the result is the corresponding control character. (That is, the character formed by taking the six least significant bits of the binary representation of the alphabetic character.) For example:

^C becomes CTRL/C

If the character following the circumflex is another circumflex, the result is a single circumflex. For example:

^^C becomes ^C
^^^C becomes ^CTRL/C

Some of the special characters are also affected by the circumflex and result in the corresponding control character. They are:

opening bracket ([),
reverse slant (\),
closing bracket (]) and
underscore (_).

NOTE

The circumflex itself is not among the listed special characters and CTRL/^ cannot be generated with the above-mentioned method.

Any combination other than those described above has no effect on the circumflex or the character following it.

4.2 LABELS

The control file can also contain labeled lines. Labels are used to identify particular lines in a control file so that they can be referenced. A label is from one to six characters in length. The first character in a labeled line is alphabetic and can be immediately followed by up to five additional alphanumeric characters. The label is terminated by two colons (::). A single line in a control file must not contain more than one label.

NOTE

Upper and lower case alphabetic characters are equivalent.

Characters following the two colons are treated as if they were in the "first character" position as described in the preceding section. In normal processing, the label itself is skipped (it does not affect the processing, but is merely a reference point), and the remainder of the line is treated as if the label were not there.

NOTE

There are two exceptions to the 6-character labels. These are the labels used in associated with the @CHKPNT and @REQUEUE commands. (Refer to Section 4.4.) Labels used with these two commands may not be more than five characters.

%ERR::, %TERR::, and %FIN:: are labels whose meanings are predefined. They cannot be referenced by commands such as @GOTO (refer to Section 4.4). These three labels are reserved for use upon the occurrence of an error indication as described below.

4.3 ERROR PROCESSING

The Batch Controller tests for error indications in lines of output sent from user programs, from system programs, and the system. The error signal is the question mark (?) appearing as the first character of an output line. An additional character can be declared by using the Batch command, @ERROR. (Refer to Section 4.4.)

When the Batch Controller detects an error indicator, it waits for the job to request more input (that is, the job has completed its current processing and is ready for another command or it has returned to system command level. At this point, error processing begins. If the error was ?TIME LIMIT EXCEEDED, special handling is involved (see Section 4.3.1). For all other errors detected, the Batch Controller bypasses all lines indicated to be user level input, and it examines the next line in the control file to be interpreted as either a system or Batch command. If this line contains an @IF command, it is processed and the error state is cleared.

NOTE

The Batch Controller does not search past the next executable system command in the control file for the @IF command. Therefore, if this command is used, there must be no intervening system level commands and it must be the next Batch command in the control file.

4.3.1 Reserved Labels

"TIME LIMIT EXCEEDED" is one type of error that can occur and is handled as follows. When the ?TIME LIMIT EXCEEDED message is issued, BATCON gives the job an additional amount of time so that it can terminate its

processing gracefully. This extra time is normally 10% of the job's original time. (The value 10% is an installation option.) Next, a search is initiated for a %TERR:: label and, if it is found, processing resumes at that label. If the end of the control file or a %FIN:: label is encountered during the search for a %TERR:: label, then the appropriate error message is issued (i.e., BTNECF or BTNFFS) and processing is continued from there. (See Chapter 5 for explanations of the BATCON error messages.)

If the time limit is exceeded again during the 10% extra, no further time is allotted, and the job is terminated.

When an error other than ?TIME LIMIT EXCEEDED is detected, the Batch Controller determines whether or not the next command is an @IF command. If it is not, the Batch Controller will then search for an error label. The label %ERR:: begins an error "packet" that is to be executed when an error occurs. An error "packet" can contain only one command or as many as the user wishes. These error "packets" may be placed anywhere in the control file. Once the commands in the "packets" have been processed, the Batch Controller continues from that point in the control file; it does not read backwards over sections of the control file that it may have skipped during its search for error "packets".

When a % type error label is encountered, the resultant action depends on whether or not an error condition exists. Non-error labels have no effect on the job; they are merely reference points. If an error condition exists at the time a non-error label is encountered, then the action depends on the contents of the line. The figures below are intended to illustrate what action is taken under these various conditions. Figure 4-1 shows the path taken under normal processing, and Figure 4-2 shows the action taken during error recovery.

When the user bypasses SPRINT and creates his own control file, he may place a %FIN:: near the end of the control file. SPRINT, in creating the control file, automatically places a %FIN:: at a point in the control file corresponding to the end of the user's input deck. This label is used for cleanup purposes; e.g., deleting the input files. In creating the control file, the user may place other %FIN::'s at various points in the file for periodic cleanup of his job. For example, this label is used in a special kind of error recovery. If the time allocated to the job runs to the maximum limit specified on the $JOB card (refer to Section 3.1) or by the Batch System, BATCON searches forward through the control file for a %TERR:: or a %FIN:: label (whichever it finds first). If either label is found, the job is given an additional 10% of its allocated time and processing resumes from that point. If the user includes a %FIN:: in his control file and an error occurs on processing the job, he gets an opportunity to execute some additional commands to clean up his job rather than having it terminated in an unknown state.

NOTE

The extra time is given only once for a job and if this additional time is exceeded, the job is cancelled.

The user should be careful in placing the %FIN:: in the control file because if the Batch Controller is searching for an error recovery routine and %FIN:: is placed before a %ERR:: or %TERR::, the %FIN:: is executed and the Batch Controller assumes the error recovery routine has been satisfied and does not search any longer for %ERR:: or %TERR::. Furthermore, an @GOTO label cannot bypass a %FIN:: label. Therefore, the best place to put a %FIN:: is near the end of the control file.

If an error occurs in the job and the user has not included an @IF command, the Batch Controller searches for %ERR::. Further processing depends on the success or failure of this search, as follows.

1. If %ERR:: is found, the error "packet" is processed.
2. If %FIN:: is found, the Batch Controller resumes processing from that point.
3. If the end of the control file is reached before either %ERR:: or %FIN:: is encountered, the Batch Controller issues an error message and the job is terminated.

When the Batch Controller detects an error in the Batch commands entered by the user, control is transferred to %FIN::. Refer to Chapter 5 for the Batch Controller messages.

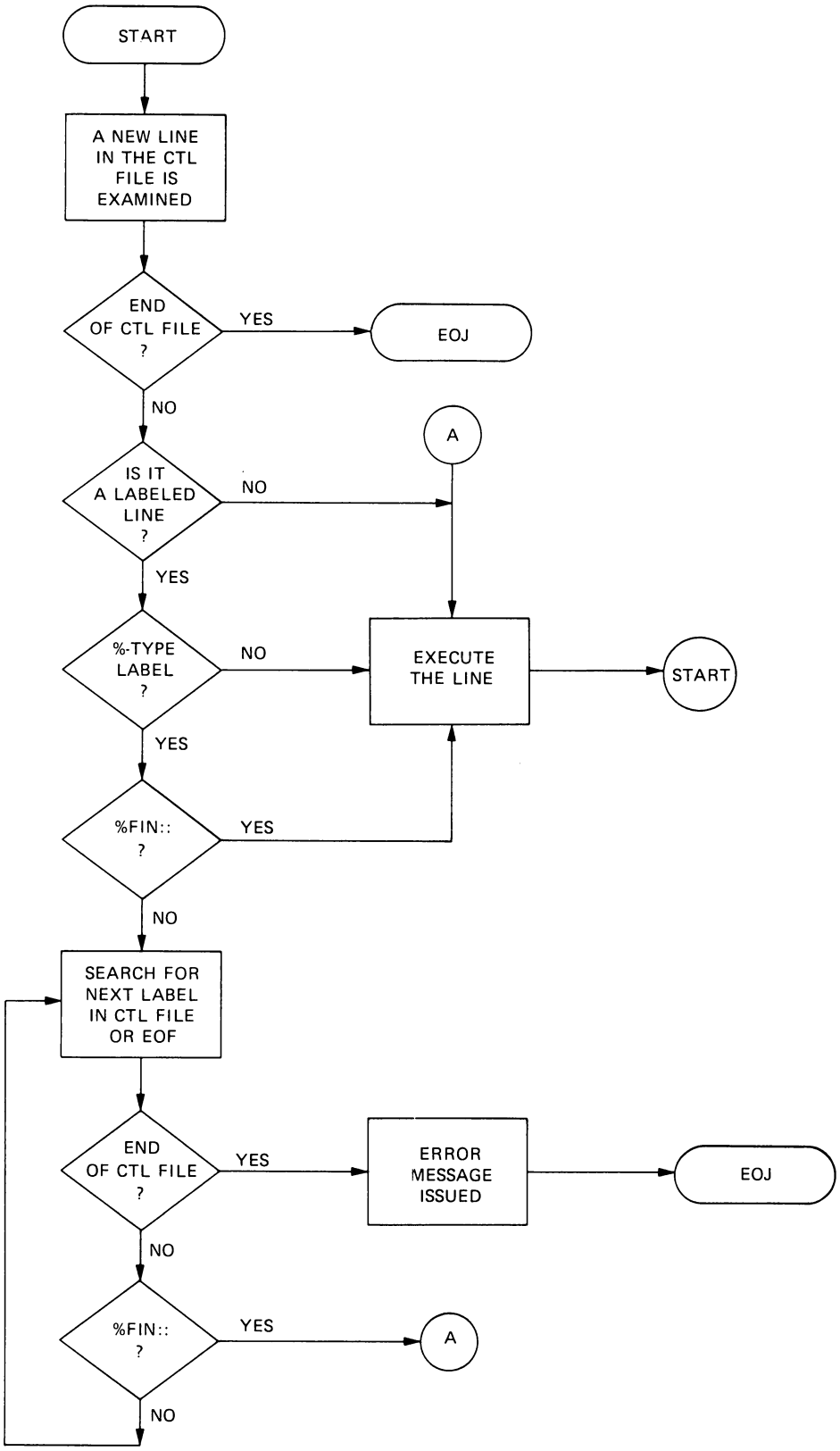


Figure 4-1 Normal Processing

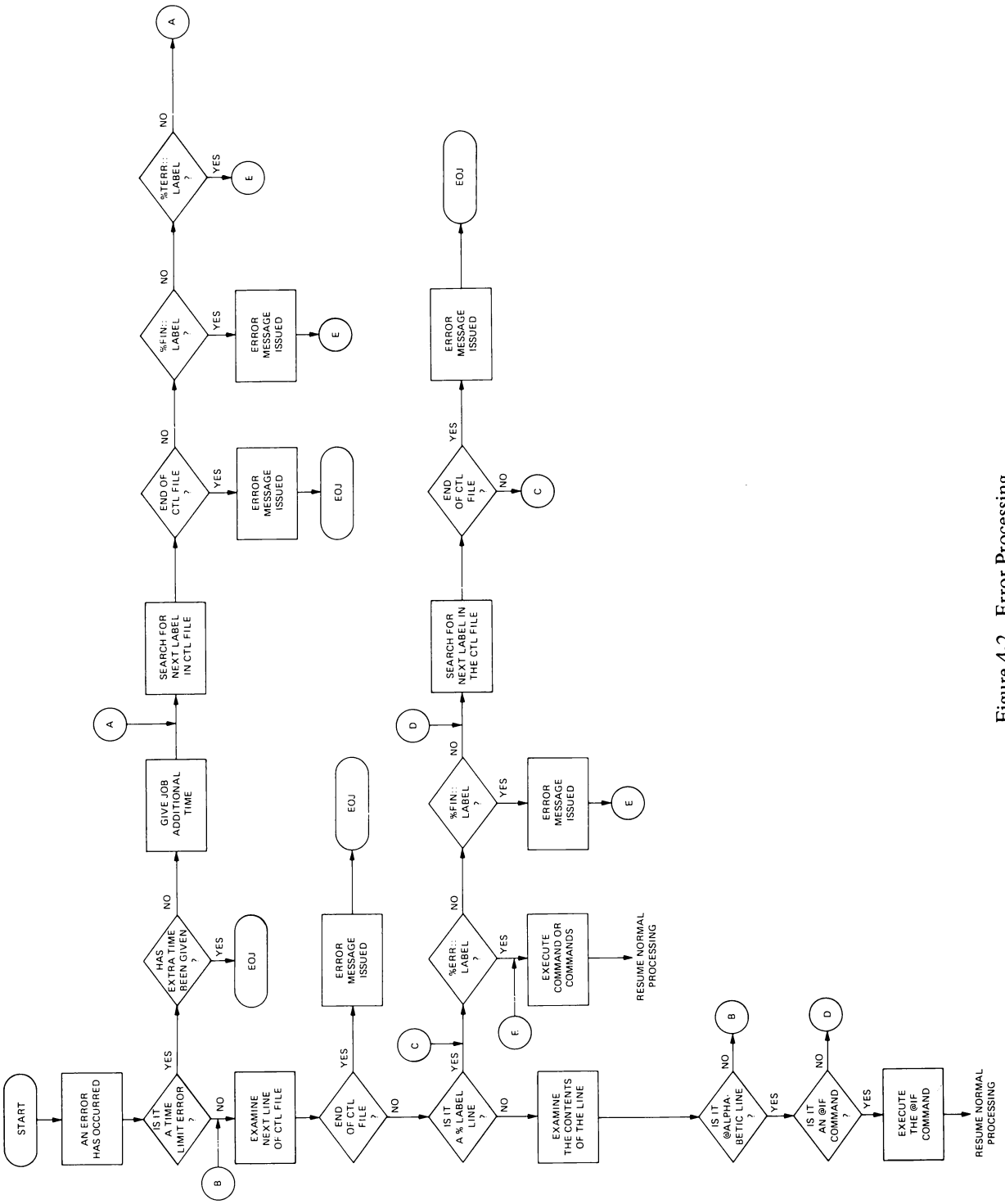


Figure 4-2 Error Processing

4.4 BATCH CONTROLLER COMMANDS

The Batch Controller examines lines beginning with an at sign to determine whether the line is a Batch command or a system command. If it is a system command, the job is placed at system command level before the command is sent to the system for action. An important difference between these two types of commands is that Batch commands do not affect the running job.

Comments may appear on lines containing Batch commands. Comments are identified by an exclamation point (!) appearing after the command and before the text to be treated as a comment. For example:

```
@IF (ERROR) @GOTO A     !SKIP THE FOLLOWING IF ERRORS
                        .
                        .
                        .
A::   !BEGIN OWN ERROR ROUTINE
                        .
                        .
                        .
```

Batch commands may be abbreviated to as few letters as are necessary to uniquely identify a command. However, a minimum of two letters is required since the system has single letter commands that could be construed as Batch commands (e.g., G for GET and GOTO). Any command that is an ambiguous abbreviation, a single letter, or not one of the following Batch commands, is transmitted to the system. If it is not a valid system command, the system will issue an error message, and the Batch Controller will initiate error processing.

@BACKTO

Function

The @BACKTO command is used to direct the Batch Controller to search back in the control file for a line with the specified label.

The search starts back at the first line of the control file and continues until the specified label is found. At this point, the Batch Controller transfers control to the statement associated with the label. If the label is not found, the job is terminated.

Command Format

@BACKTO *label*

label = a 1- to 6-character label as defined in Section 4.2.

NOTE

If an @BACKTO command occurs after a %FIN:: label in the control file, the label specified in the @BACKTO command may also be after the %FIN::.

Since the system has a BACKSPACE command, the only abbreviation of @BACKTO that will be recognized is @BACKT (i.e., "BACK label" will be considered a TOPS-20 BACKSPACE command and it will be passed to the system).

@CHKPNT

Function
This command directs the Batch Controller to save a potential restart point to be used in the event of a system failure. As many @CHKPNT commands as desired may be placed in the control file. If the job is restarted, processing will begin at the label specified in the last @CHKPNT command executed before the restart.

Command Format
@CHKPNT *label*

label = a 1- to 5-character label as defined in Section 4.2.

The @CHKPNT command can affect the restarting of a job after a system failure. If no restart was specified for the job, and there was a system failure, and if the job did not execute an @CHKPNT command, then the job will not be restarted. If restart was specified, and the system failed, but no @CHKPNT command was executed, then the job will be restarted at the beginning of the control file. If the job did execute an @CHKPNT command and there was a system failure, then the job will be restarted regardless of the restart specification. The job will be restarted at the first occurrence of the label specified in the most recently executed @CHKPNT command.

NOTE

1. The restart parameter may be specified on the $JOB card in SPRINT (refer to Section 3.1).or in the SUBMIT command (refer to the *User's Guide*).
2. The label specified for an @CHKPNT may occur after a %FIN:: label.

@ERROR

Function
This command causes the specified character to be recognized as an error signal by the Batch Controller. The question mark (?) is always recognized as an error indicator irrespective of the use of the @ERROR command.

Command Format
@ERROR *character*

character = the beginning character of the line that is to be recognized as an error. If omitted, only the question mark (?) will be recognized.

The character specified may not be a control character or an exclamation point (!). The exclamation point will be interpreted as the comment character and will not function as the error signal character.

@GOTO

Function
The @GOTO command directs the Batch Controller to search the control file in a forward direction for the specified label. When the label is found, control is transferred to the statement associated with the label. If the label is not found, the search will be satisfied if a %FIN:: is encountered. If neither the label for a %FIN:: is found by the time

the end of the control file is reached, the Batch Controller will issue the error message BTNCNF (refer to Chapter 5 for the Batch Controller messages) and the job will be terminated.

Command Format
@GOTO *label*

label = a 1- to 6-character label as defined in Section 4.2.

NOTE
The search initiated by the @GOTO command cannot by-pass a %FIN::.

@IF

Function
This command directs the Batch Controller to test for the condition specified in the command string. If the condition is true, the object statement is executed. Otherwise, this command and its object statement are treated as comments and the Batch Controller proceeds to the next line in the control file.

Command Format
@IF *(condition)statement*

(condition) = (ERROR) – determine if an error signal has been issued.

or

(NOERROR) – determine if no error signal has been issued.

The parentheses must be included.

statement = this parameter is optional but, if present, can be a Batch or system command, user or system program data, or a comment.

The @IF command is used to assist the Batch user in processing errors. The Batch Controller recognizes the existence of an error when it encounters a line beginning with a question mark (or with the character specified in the @ERROR command) that is output from the job. When an error occurs, the @IF command must be the next Batch command in the control file. There must be no intervening system commands.

NOTE
The @IF command cannot be used to intercept a ?TIME LIMIT EXCEEDED error.

@MESSAGE

Function
When SPRINT encounters a $MESSAGE card (refer to Section 3.1), it creates an @MESSAGE command in the control file. Direct use of the @MESSAGE command is strongly discouraged (because of possible compatibility problems with future versions of the Batch System) and for this reason, the syntax for the command is not specified here.

@NOERROR

Function
The @NOERROR command instructs the Batch Controller to ignore all error signals except the ?TIME LIMIT EXCEEDED message and any error issued by the Batch Controller itself. (Refer to Chapter 5 for Batch Controller messages.)

Command Format
@NOERROR

@NOOPERATOR

Function
This command directs the Batch Controller to terminate the dialogue mode between the job and the system operator. (Refer to Section 4.5.)

Command Format
@NOOPERATOR

@OPERATOR

Function
This command directs the Batch Controller to initiate the dialogue mode between the user's job and the system operator. (Refer to Section 4.5.)

Command Format
@OPERATOR *character*

character = The character that begins a line that is to be sent to the operator (e.g., #). Specification of the character is optional and if it is omitted, the dollar sign ($) will be assumed to be the operator signal character.

NOTE
The character must *not* be a control character, a semicolon (;) or an exclamation point (!).

@PLEASE

Function
This command directs the Batch Controller to type the specified message to the system operator. The operation of the @PLEASE command is not affected by the @OPERATOR and @NOOPERATOR commands.

Command Format
@PLEASE *message* ESCape (CR/LF)

message = the message to be typed to the system operator.

ESCape = the ESCape character. If this character is present, processing continues normally after the message has been output to the operator. If the character is omitted, the job will wait for a response from the operator before resuming its normal processing. The ESCape character may be generated via the sequence ^[(circumflex-opening bracket).

CR/LF = the carriage-return/line-feed is required.

@REQUEUE

Function
The @REQUEUE command indicates to the Batch Controller that the job is to be requeued for processing at a later time.

Command Format
@REQUEUE *label*

label = a 1-to 5-character label as defined in Section 4.2.

When the job is requeued, processing will continue at the specified label. If the label is omitted, the job will be requeued at the label specified in the last @CHKPNT command executed before the @REQUEUE. If the label is omitted and no @CHKPNT command has been executed, the job will be restarted at the beginning of the control file.

NOTE
The label specified for an @REQUEUE may be after a %FIN:: label.

@REVIVE

Function
The @REVIVE command directs the Batch Controller to resume normal listing in the LOG file (i.e., all output from the job is placed in the LOG file). The initial state of the Batch Controller is to list all output and the @REVIVE command is used to clear the effect of the @SILENCE command.

Command Format
@REVIVE

@SILENCE

Function
The @SILENCE command directs the Batch Controller to suppress all input and output except error messages from the LOG file. This suppression of output is cleared by the @REVIVE command.

Command Format
@SILENCE

NOTE
Batch commands are always printed in the LOG file.

4.5 OPERATOR – PROGRAM COMMUNICATION

The Batch Controller provides capabilities for communication between a running program and the system operator. Two methods for this type of communication are available. With the first method, it is possible for the Batch Controller to enter a conversational or "dialogue" mode (refer to paragraph 4.5.1) with the system operator. The second method provides a facility for the program to send a 1-line comment to the operator (refer to paragraph 4.5.2).

4.5.1 Dialogue Mode

When dialogue mode is active (refer to paragraph 4.5.1.1), it is possible for a running program to "converse" with the system operator. In this mode, lines of output from the program are entered on the terminal controlling the Batch Controller (usually the system operator's terminal) as well as entered in the job's LOG file. Requests for input (initiated by the job) are handled from the same terminal, rather than from the control file. Whether the input entered by the operator is interpreted at system level or user level is determined by the level at which input is requested. The operator cannot enter Batch Controller commands, such as @GOTO, to alter the sequence of the user's control file.

There are two commands associated with dialogue mode: @OPERATOR, which initiates dialogue mode and @NOOPERATOR, used to terminate dialogue mode.

4.5.1.1 The @OPERATOR Command – This command causes the Batch Controller to enter dialogue mode. (Refer to Section 4.4 for a description of the command format.) While BATCON is in this mode, any line of output from the job that begins with the character specified in the command is entered on the controlling terminal as well as written in the job's LOG file. In addition, all subsequent lines, regardless of their initial character, are copied in the same manner. This copying continues until the job requests input or it returns to the system (the system requests input).

At this point, rather than reading the next line from the control file, the Batch Controller notifies the operator of the request, and the job is suspended until the operator has responded. When the operator responds, the job is resumed and the response is sent to the job as if it had appeared on the control file. The LOG file contains this line and it is identified as an operator response. Also, output to the operator is discontinued until the job outputs another line beginning with the specified character. The process described above is then repeated.

4.5.1.2 The @NOOPERATOR Command – This command causes the Batch Controller to terminate dialogue mode. (Refer to Section 4.4 for a description of the command format.) The @NOOPERATOR command is in effect at the time BATCON is initialized.

4.6 BATCH CONTROLLER OUTPUT

As part of the Batch job's output, a transaction log is produced by the Batch Controller. This file contains a record of each line of the control file as passed to the job, every line of output sent by the user job, Batch Controller messages, and any operator actions.

4.6.1 The LOG File

The contents of the LOG file appear exactly as they would have if the user had been running at a timesharing terminal, with the following exception: each line begins with a time stamp to indicate the time at which the line was processed and a notation to indicate the type of the line.

The time stamp has the format hh:mm:ss to indicate the time of day on a 24-hour clock. The notation is next, followed by the line of input or output. The notation is one of the following:

BAJOB	identifies the start of a new job. Information on this line is the job name, sequence number, user's name, and internal stream number.
BAFIL	contains the file specification for the control file or the LOG file.

BLABL	identifies a label found in the control file. The rest of the line is listed in the LOG file as a separate line.
BATCH	indicates a Batch Controller command that was processed.
TRUE	indicates that the condition tested in the Batch Controller command @IF was true. The object statement is listed in the LOG file on a separate line.
FALSE	indicates that the condition tested in the Batch Controller command @IF was false. The object statement is listed as a comment on the same line with the @IF statement.
BAOPR	identifies a line entered by the system operator. Further identification carries the specific operator command and comments from the operator. This identification can be TELL, KILL, STOP, GO, and OPERATOR.
MONTR	indicates that the output line was from the system or that the input requested was to be processed as a system command.
USER	indicates that the output was sent from a user or system program or that the input requested was to be sent to a running program.
BASUM	indicates a summary line of the job parameters.
BAUSR	indicates that the job was cancelled at the user's request.

Any lines in the LOG file that do not appear as described above are comments, error messages from the Batch Controller, or lines of the control file that were skipped because of errors.

4.6.2 Batch Controller Messages

Messages issued by the Batch Controller are entered into the LOG file as comments. Blank lines surrounding the message isolate it for convenient recognition. In addition, each message issued by the Batch Controller has a 6-letter prefix of the form BTN*xxx*. BTN identifies it as a message from the Batch Controller and *xxx* represents a 3-letter abbreviation of the text of the message. For example:

```
BTNJRQ Job Requeued
```

A complete list of the Batch Controller messages can be found in Chapter 5.

CHAPTER 5
ERROR MESSAGES

The following conventions are used in describing the Batch system messages:

$cardname	a control card name
dev	a device name
file.typ	a filename and file type
n	a decimal number
x	an alphanumeric character
/switch	a switch name

Most messages returned to the user fall in one of three categories. These categories are determined by the beginning character of the message.

? at the start of the message indicates a fatal error message.

% at the start of the message represents an advisory or warning message.

[at the beginning of the message indicates a comment line.

In addition, each message issued has a 6-letter prefix. The first three letters indicate the program that issued the message. The last three letters represent an abbreviation of the text of the message. For example:

?LPTPLE PAGE LIMIT EXCEEDED

5.1 BATCON MESSAGES

BTNBLA BEGINNING PROCESSING AT LABEL *xxxxx*

The job is restarting at the indicated label. This is an advisory message which occurs after a system reload when the user has specified a CHKPNT label and the CHKPNT has been taken. (Refer also to the description of the /TAG: switch for SUBMIT in the *User's Guide.*)

BTNBLI BEGINNING PROCESSING ON LINE *n*

The job is restarting at the specified line in the control file.

BTNBPL YOUR USE OF BACKTO HAS CAUSED A POSSIBLE LOOP

It is possible that the user has created a loop in his control file through improper use of the BACKTO command. For example, A::@BACKTO is a loop and would generate this message.

BTNCFC CANNOT FIND THE CTL FILE. ERROR CODE = *xxxxxx*

The specified control file could not be found on the disk structure specified. The notation "*xxxxxx*" is the error code returned by the GTJFN or OPENF monitor call. (Refer to Appendix A of the *DECsystem-20 Monitor Calls Reference Manual.*)

BTNCNF COULD NOT FIND LABEL *xxxxxx*

The label specified in an @GOTO or an @BACKTO command could not be found in the control file. A common occurrence of this error is:

@GO TO *label*

There is a space between GO and TO. GO is a valid abbreviation for GOTO, and the label specified is TO. Another example is:

@BACKTO ABCD
ABCD::*@command*

The label specified in the BACKTO command must be *before* the occurrence of the BACKTO command in the control file.

BTNCRL CHKPNT OR REQUEUE LABELS CANNOT BE OVER 5 CHARACTERS LONG

The label specified in a CHKPNT or REQUEUE command was greater than five characters in length. The maximum number of characters for the labels associated with these two commands is five.

BTNECF END OF THE CONTROL FILE WHILE SEARCHING FOR *%xxxx*

The end of the control file was reached while the Batch Controller was searching for *%xxxx*, where *xxxx* is one of the error recovery labels (%ERR::, %TERR::, or %FIN::).

BTNFFS FOUND %FIN WHILE SEARCHING FOR *xxxxxx*, PROCEEDING FROM %FIN

A %FIN:: was encountered during a search initiated by an @GOTO command or an implicit @GOTO (searches for %ERR:: or %TERR::). %FIN:: takes precedence during searches for %ERR::, %TERR::, or a label specified in an @GOTO command. The *xxxxxx* is the label the Batch Controller was searching for at the time it encountered the %FIN::.

This message could be issued instead of either BTNCNF or BTNECF, depending on the construction of the control file.

BTNICS ILLEGAL CHARACTER SPECIFIED FOR *xxxxxx*

The user specified a character that is an illegal signal character for *xxxxxx*, where *xxxxxx* is either ERROR or OPERATOR. The signal character for these two commands must not be a control character, an exclamation point (!) or a semicolon (;).

BTNIIC ILLEGAL IF COMMAND OR SYNTAX ERROR

The operand of the @IF command could not be recognized, or the format of the @IF command is invalid.

BTNJBC JOB HAS BEEN CANCELLED

The system would not allow this job to log in. The reason is specified in the output from the LOGIN program.

BTNJNR JOB CANCELLED AFTER RESTART, IT IS NOT RESTARTABLE.

The job has a RESTART parameter of NO (0), and it has been reselected after a system reload. In accordance with the definition of /RESTART, the job is being cancelled.

BTNJRQ JOB REQUEUED

The user's Batch job has been requeued. Other messages accompany this to indicate the reason.

BTNNLS NO LABEL SPECIFIED OR ILLEGAL SYNTAX

The label was not specified for a command that requires one, or a punctuation error was made.

BTNSND CTL FILE STRUCTURE IS NOT A DISK. DEVICE IS *dev*:

The user specified a device *dev*: that is not a disk. The control file must reside on a disk.

5.2 LPTSPL MESSAGES

?LPTCAF CAN'T ACCESS FILE *xxx*. CODE *nnn - xxx*

LPTSPL encountered an error while looking up the file it was trying to print. The *nnn* stands for the number and *xxx* for the text of one of the error codes listed in Appendix B.

?LPTCBU JOB CANCELLED ON REQUEST BY USER: *xxx*

User *xxx* cancelled the job from his terminal.

[LPTCPT CHECKPOINT TAKEN DURING FILE *xxx*, COPY *nnn*, PAGE *nnn*]

LPTSPL took a checkpoint at page *nnn* of copy *nnn* of file *xxx*.

[LPTFPF FINISHED PRINTING FILE *name*]

This message appears in the user's Batch log file to inform him that LPTSPL completed processing the file *name*.

?LPTFSD FILE SKIPPED DUE TO I/O ERRORS, STATUS *nnn*

LPTSPL skipped the file it was processing because it encountered too many I/O errors while it was reading the file it was trying to output, LPTSPL made an attempt to read the file n times where n is a number set by the system administrator at assembly time.

[LPTJBS JOB *name*, SEQUENCE *n* STARTED AT *date-time*

This message appears in the Batch LOG file to indicate the time that LPTSPL started processing the specified job.

?LPTKBO JOB KILLED BY THE OPERATOR

The operator killed the job.

[LPTOCL OPERATOR CHANGED LIMIT TO *nnn* PAGES]

The page limit for jobs on the line printer was changed to *nnn* by the operator.

[LPTORC OPERATOR REPRINTED 1 COPY OF *xxx*]

The operator reprinted one copy of file *xxx*. The operator generally does this in instances where the original copy is illegible for some reason (e.g., the printer jammed).

[LPTOSC OPERATOR SKIPPED 1 COPY OF *xxx*]

The operator skipped one copy of file *xxx*. This usually occurs when there is some difficulty in printing a particular copy of a file (e.g., a paper jam).

[LPTOSF OPERATOR SKIPPED FILE *xxx*]

The operator skipped printing file *xxx*. One reason he might do this is that LPTSPL is unable to read the file. Normally the operator does not skip the file until several attempts have been made to read it.

?LPTPLE PAGE LIMIT EXCEEDED

The number of pages printed by this job has exceeded the page limit specified for the job.

[LPTRBO JOB REQUEUED BY THE OPERATOR]

The job was requeued by the operator. The operator might do this because of a paper jam or because it might be more convenient to print the job later.

[LPTSTF STARTING FILE *name*]

This message appears in the Batch LOG file to indicate that LPTSPL started processing the file *name*.

5.3 QUENCH MESSAGES

?QUEBSI /BEGIN IS ALLOWED FOR INP AND LPT ONLY

The /BEGIN switch can be used only with the SUBMIT (or QUEUE INP:) command or with the PRINT or (QUEUE LPT:) command.

?QUECDO CAN NOT DO OUTPUT TO DEVICE

QUENCH is unable to output queue listings on the specified device. Either the device specified does not exist or it is not available.

%QUECPB CAN'T PRINT BINARY *filespec*

The user attempted to print the binary file *filespec* with the switch /PRINT:ASCII specified (either explicitly or implicitly). Note that /PRINT:ASCII is the default.

?QUEDMI /DEFER REQUEST MUST INCLUDE /CREATE OR /KILL

The /DEFER switch must be used in conjunction with either the /CREATE or the /KILL switch.

?QUEDND INPUT DEVICE NOT A DISK *dev*:

An input device (*dev*:) specified in the QUEUE command is not a disk. The QUENCH program can handle files only from disk.

?QUEFND CONTROL AND LOG FILES MUST BE DISTINCT

The user must specify two separate files for the control and LOG files in his SUBMIT (or QUEUE INP:) command. The filenames may be the same if the types are different (e.g., JOB1.CTL, JOB1.LOG).

?QUEFRI FILENAME REQUIRED FOR INPUT QUEUE

The user must specify a filename for his control file.

?QUEICF ILLEGAL TO CREATE REQUEST FOR SOMEONE ELSE

The user may not create a request for another user. For example, the command @SUBMIT [10,33] = JOB.CTL would generate this message.

?QUEINF INCORRECT NUMBER OF FILES IN INPUT QUEUE REQUEST

Only two files may be given in a SUBMIT (or QUEUE INP:) command. These files are the control file and the LOG file specifications.

?QUEIQN ILLEGAL QUEUE NAME

The name of the queue specified in the QUEUE command is not a valid queue name. Valid queue names are: INP: and LPT:.

?QUELFS LOGIN PLEASE TO USE SWITCH *xxx*

The user must be logged in to run the QUENCH program with the *xxx* switch, where *xxx* represents /CREATE, /MODIFY, or /KILL.

?QUELOF LISTING OPEN FAILURE ON DEVICE

The output device specified for a queue listing either does not exist or is not available.

?QUELQU LISTED QUEUE USER CANNOT INCLUDE SFDS

The user cannot ask for a list of queues by SFD's. For example, the command PRINT [30,54,A]/1 is illegal and would generate this message.

?QUELTL LOGIN TO LIST ON DEVICE

The user must be logged in before he can write queue listings on a device other than a TTY.

?QUEMDC MOUNTABLE DEVICE COMMANDS NOT IMPLEMENTED

The version of QUENCH that is currently running is unable to handle mountable device commands.

?QUENAF NOT ALL REQUESTED FILES EXIST

Some of the files in the QUEUE request do not exist and the user specified the /ALLFILES switch.

?QUENFI NO FILES IN REQUEST

The user did not specify any files in his QUEUE request or none of the files he did specify exist.

?QUENQS NO QUEUE SPECIFIED IN /KILL OR /MODIFY

The user must specify which queue contains the request he wants to kill or modify.

?QUENSD INPUT DEVICE DOES NOT EXIST *dev*:

The user specified an input device (*dev*:) that does not exist. The device may have been misspelled or it might not be on-line.

?QUENSR JOBNAME OR /SEQUENCE REQUIRED

The user must specify either the jobname or sequence number (or both) of the request(s) he wants to modify or kill.

?QUEOEI OUTPUT EXTENSION IS ILLEGAL

The jobname must not have an extension (i.e., a file type).

%QUEOTE INPUT QUEUE REQUEST USES ONLY TWO ENTRIES

Only two files may be specified in a SUBMIT (or QUEUE INP:) command. The two file specifications are for the control and log files.

?QUEOWI OUTPUT WILDCARDS ARE ILLEGAL

The jobname specification on a QUEUE/CREATE command and the output specification on a QUEUE/LIST command cannot have wildcard constructions.

%QUEPSI /PROTECT:777 IS ILLEGAL, CHANGED TO 677

A protection code of 777 for a queue request is not permissible. The protection was changed to 677.

?QUEPSI PAPER SWITCH ILLEGAL

The user cannot specify the /PAPER switch on the SUBMIT (or QUEUE INP:) command.

[QUEQLF QUEUE LISTING ON FILE *filespec*]

The user requested a listing of one or more queues to the specified file *filespec.*

?QUEROM /REMOVE IS LEGAL ON MODIFY REQUESTS ONLY

The /REMOVE switch must be used in conjunction with the /MODIFY switch.

?QUESJN SPECIFY JOBNAME LEFT OF EQUAL SIGN

The jobname must be to the left of the equals sign.

?QUESVT SWITCH VALUE TOO LARGE

The user specified a value on the /DEPEND switch that is too large.

?QUETCT TIME COMPONENT TOO LARGE

The amount of time specified on the /AFTER or the /TIME switch is too large. The value of hh:mm:ss for the /AFTER switch cannot be greater than 23:59:59. The value for the /TIME switch cannot be greater than 63:59:59.

?QUETTL TIME TOO LARGE

A time field contains a value which is larger than the maximum permissible.

?QUEUCI QUEUE USER CANNOT INCLUDE SFDS

SFDs may not be included as part of the jobname specification.

?QUEWDI WILDCARD ILLEGAL IN INPUT QUEUE FILE DIRECTORY

A wildcard construction is not permissible in an input queue file specification. For example, the command SUBMIT JOB.CTL[40,*] is illegal and would generate this message.

?QUEWIE WILDCARD ILLEGAL IN INPUT QUEUE FILE EXTENSION

The user must not specify a wildcard extension (i.e., file type) in his SUBMIT (or QUEUE INP:) command. The command SUBMIT JOB.* is illegal.

?QUEWIN WILDCARDS ILLEGAL WITH /NEW

The use of /NEW implies that the files have not yet been created. Therefore, it is not possible to perform a wildcard search for the file.

?QUEWIQ WILDCARD ILLEGAL IN INPUT QUEUE FILE NAME

The user must not specify a wildcard filename in his SUBMIT (or QUEUE INP:) command. The command SUBMIT *.CTL is illegal.

5.4 SPRINT MESSAGES

?SPTABO JOB ABORTED BY OPERATOR

The operator intentionally terminated the job.

?SPTAMO ASCII MODE ONLY ON NON-CDR DEVICES

The user has attempted to read a non-ASCII file on a device other than the card reader.

%SPTBDT BAD DATE-TIME SPECIFICATION ON */switch* SWITCH – IGNORED

The specified switch will be ignored because the date-time was specified in an invalid or unrecognizable format.

?SPTBMN BATCH JOB MAY NOT RUN IN A FILES ONLY DIRECTORY

The user has attempted to create a Batch job but the directory specified on the $JOB card is a files-only directory (one in which no LOGINs are permitted).

[SPTCBO JOB CONTINUED BY OPERATOR]

The operator resumed the processing of the job. This message would occur subsequent to [SPTSBO JOB STOPPED BY OPERATOR].

?SPTDDI DOUBLE DIRECTORY ILLEGAL

Two directory names cannot appear without an intervening filename.

?SPTDDV DOUBLE DEVICE ILLEGAL

Two device names appeared in a row without an intervening filename (e.g., LPT:PTP:) or two colons appeared in a row (e.g., DSK::FOR1.DAT).

%SPTDEX DOUBLE EXTENSION ILLEGAL

Two extensions (i.e., file types) cannot appear in a row without an intervening filename or comma (,).

?SPTDFN DOUBLE FILENAME ILLEGAL

Two filenames appeared in a row (e.g., $COBOL TEST1 TEST2) or two periods appeared in a row (e.g., TEST3..CBL).

?SPTDNA DEVICE *dev* NOT AVAILABLE

The specified output device cannot be initialized because it is currently being used or because it does not exist.

?SPTDND DEVICE *dev* IS NOT A DISK

The user asked to spool files onto a device that was not a disk.

%SPTEPF EXTRANEOUS PASSWORD CARD FOUND – IGNORED

Either the installation does not require a password, or the user included more than one $PASSWORD card with his job.

?SPTEWF ERROR WRITING FILE

An I/O error occurred while the user's file was being written.

[SPTFOD FILES ONLY DIRECTORY

The directory specified in the $JOB card is a files only directory; the user may not log in to it.

?SPTFSR FILESPEC REQUIRED ON $INCLUDE CARD

The user has omitted the file specification on the $INCLUDE card. It is necessary to correct the card and start the job again.

[SPTFST FILE SUBMITTED TO LPT QUEUE]

This message generally is issued in response to a $CREATE card with the /PRINT switch specified.

%SPTHOL *n* HOLLERITH ERRORS IN CARD # *n*

SPRINT detected *n* Hollerith errors in card number *n*.

%SPTIBC ILLEGAL BINARY CARD – CARD # *n*

Card number *n* is not a legal binary card.

?SPTICC ILLEGAL CONTROL CARD $*cardname* – CARD # *n*

The card name on card number *n* is not a name recognized by SPRINT. Refer to Chapter 3 for a description of the SPRINT control cards.

?SPTICS ILLEGAL COMMAND SWITCH ON THE $*cardname* CARD

Either the switch specified for the command was not a valid switch for that command or the format was incorrect. A *Command Switch* must immediately follow the command with which it is associated with no intervening blanks. Refer to Chapter 3.

?SPTIDP INCORRECT DIRECTORY OR PASSWORD

The user specified a directory which does not exist or he supplied a password that is invalid for the specified directory.

?SPTIDS ILLEGAL DIRECTORY SPECIFICATION

The format of the directory information enclosed within the square brackets ([ ]) or angle brackets (< >) is incorrect.

%SPTISW */switch* IS NOT LEGAL ON THE $*cardname* CARD – IGNORED

The switch specified by the user is not valid on the SPRINT control card. The switch will be ignored. Refer to the individual control card descriptions in Chapter 3 to determine the switches that are valid for each card.

?SPTLKE LOOKUP/ENTER ERROR *n* – *xxx*

While creating a file for the user, SPRINT encountered an error. The *n* represents the number and *xxx* represents the text of one of the messages in Appendix B.

%SPTMSV MISSING SWITCH VALUE ON THE */switch* SWITCH – SWITCH IGNORED

A switch was specified but no value was given for it. The job will be processed as it would have been had the switch not been specified.

?SPTNDV NULL DEVICE ILLEGAL

A colon has been found without a preceding device name.

%SPTNEF NO END-OF-FILE CARD FOUND – EOF ASSUMED

SPRINT encountered a new $SEQUENCE or $JOB without finding a previous $EOJ card.

%SPTNFT NO FILES TO LOAD

The user has placed a $DATA or $EXECUTE card in his deck but has not preceded it by a $-*language* or a $INCLUDE card. Refer to Chapter 3.

?SPTPWR PASSWORD IS REQUIRED

The user must give a password which matches the password in the accounting files for the specified user name.

?SPTQTA QUOTA EXCEEDED ON *dev* WRITING USER FILE

SPRINT was unable to spool the user's input because of a lack of sufficient space on the specified device.

[SPTSBO JOB STOPPED BY OPERATOR]

The operator temporarily stopped the job.

%SPTSOR SWITCH VALUE OUT OF RANGE ON THE */switch* SWITCH – SWITCH IGNORED

The value specified for a switch was not within the range allowed for that switch. For example, a value greater than 80 is not valid for the /WIDTH switch. The switch will be ignored and the default value will be assumed.

?SPTTMH TOO MANY HOLLERITH ERRORS

The number of Hollerith errors has exceeded the limit specified in the /ERROR switch on the $JOB card.

?SPTUCO UNEXPECTED CHARACTER(S) "*x*" ON CONTROL CARD – IGNORED

While scanning for a switch, SPRINT encountered character *x*. SPRINT will ignore this unexpected character and will look for a slash (/) indicating the next switch or the end of card, and will ingore any information between. For example, if the user has accidentally punched a question mark (?) instead of a slash (/) to indicate a switch, then that switch will be ignored.

%SPTURS */switch* IS AN UNRECOGNIZED SWITCH – IGNORED

The switch specified is not recognized by SPRINT.

%SPTUSV UNRECOGNIZED SWITCH VALUE ON THE */switch* SWITCH – SWITCH IGNORED

The value specified is not a valid argument for the switch; either because it is not a value associated with the switch or because the format is incorrect.

APPENDIX A
SWITCH DESCRIPTIONS

Switch	Meaning	Applicable Cards or Commands
/ACCOUNT:*xxx*	This switch specifies the account to which the resources used by this job are to be billed. The argument is comprised of from 1 to 39 alphanumeric characters. **NOTE** This switch is required. If it is omitted, an error message is issued and the job is not processed.	$JOB
/AFTER: *date time*	The job cannot be run until after the specified date and time. Refer to the $JOB card in Chapter 3 for the date and time formats. The resulting AFTER time must be less than the DEADLINE time. If either the switch or the value is omitted, no AFTER constraints are assumed.	$JOB
/ASCII	The input is read in ASCII card code.	$COBOL, $CREATE, $DATA, $FORTRAN, $MACRO, $MODE, $TOPS20
/CREF	A cross-referenced listing is created to be processed by the CREF program. The default is that no CREF listing is created.	$FORTRAN, $MACRO
/DEADLINE: *date time*	The job must be completed by the specified date and time. Refer to the $JOB card in Chapter 3 for the date and time formats. The resulting DEADLINE time must be greater than the AFTER time. If either the switch or the value is omitted, no DEADLINEs are assumed.	$JOB
/DEADLINE: *+hh:mm*	The job must be started by the indicated amount of time after it is input. If either the switch or the value is omitted, no DEADLINEs are assumed.	$JOB
/DEPEND:*nn*	Specifies the initial interjob dependency count (in decimal). The default is 0.	$JOB

Switch	Meaning	Applicable Cards or Commands
/DOLLARS	When this switch is specified, and a card within the deck has a dollar sign ($) in column 1 but is not a $SEQUENCE, $JOB, $EOD, or $EOJ card, then the entire card will be treated as part of the user's data or program and not as a SPRINT control card. This switch and the /NODOLLARS switch are mutually exclusive. If /DOLLARS is not specified, then /NODOLLARS is assumed. **NOTE** Any deck for which the /DOLLARS switch is specified must terminate with one of the following cards: $SEQUENCE, $JOB, $EOD, or $EOJ.	$COBOL, $CREATE, $DATA, $FORTRAN, $MACRO, $TOPS20
/ERROR:*nn*	Specifies the number of invalid cards the job can read before the job is terminated. Where: *nn* represents the number of Hollerith errors permitted before the job is terminated. If the switch is not specified, a default value of 5 is assumed. However, if the switch is specified, but no argument is given, Batch assumes that *nn* has a value of zero.	$JOB
/IMAGE:*nn*	The card deck is read in image mode. The switch must be followed by a decimal number in the range 2 through 80. This causes ensuing cards to be read in image mode until either end-of-file is reached or a card is read that contains punches in all rows of column 1 and in all rows of column *n*. The SPRINT control commands are not recognized when cards are read in image mode. If no value is given for *n*, then SPRINT will assume 2.	$CREATE, $DATA
/JOBNAME:*xxx*	Specifies a 1 to 6 character user-assigned name for the job. If omitted, SPRINT creates a unique name of the form JB*xxx* (where *x* represents a character arbitrarily chosen by SPRINT to produce a unique name) for the control and log files.	$JOB

Switch	Meaning	Applicable Cards or Commands
/LIST	A compilation listing will be generated. /LIST is the default.	$COBOL, $FORTRAN, $MACRO
/LOGDISP:*arg*	This switch specifies the disposition of the LOG file after the Batch job has been processed. This switch has two arguments, one of which may be specified, as follows: 1. DELETE - delete the LOG file after printing it. The default is DELETE. 2. PRESERVE - save the LOG file after printing it. **NOTE** LOG files are appended to, not superseded. That is, if a job is run with PRESERVE specified for the LOG file then another job with the same name is run, the new LOG file will be appended to the already existing LOG file.	$JOB
/MAP	This switch causes a loader map to be generated and printed. The default is /NOMAP.	$DATA, $EXECUTE
/NODOLLARS	If this switch is in effect and a card beginning with a dollar sign ($) is encountered while the deck is being copied, SPRINT examines column 2. If column 2 contains a dollar sign, the initial dollar sign is suppressed, and the remaining information on the card is copied onto disk. If column 2 has an alphabetic character, the card is treated as a SPRINT control card. If column 2 is any character other than a dollar sign or an alphabetic character, then the card is treated as part of the user's data. /NODOLLARS is the default. **NOTE** Any deck for which the /NODOLLARS switch is specified (either by including the switch or by default) can be terminated with any SPRINT control card other than a $MODE.	$COBOL, $CREATE, $DATA, $FORTRAN, $MACRO, $TOPS20
/NOLIST	No listing file of the program will be created. /LIST is the default.	$COBOL, $FORTRAN, $MACRO

Switch	Meaning	Applicable Cards or Commands
/NOMAP	A loader map will not be generated. This is the default.	$DATA, $EXECUTE
/NORESTART	The job is not restartable. If this switch is omitted, it is assumed that the job can be restarted.	$JOB
/NOSUPPRESS	Trailing blanks are not suppressed. /NOSUPPRESS is the default.	$COBOL, $CREATE, $DATA, $FORTRAN, $MACRO, $MODE, $TOPS20
/NOWAIT	The job will continue after typing a message without waiting for a response from the operator. This is the default.	$MESSAGE
/OUTPUT *xxx*	This switch determines whether or not the LOG file is printed. This switch has three arguments, one of which may be specified as follows: 1. LOG - print the LOG file 2. NOLOG - suppress printing of the LOG file 3. ERROR - print the LOG file only if errors are encountered. If this switch is omitted, the LOG file will be printed.	$JOB
/PAGES:*nn*	Specifies the maximum number of pages (in decimal) to be printed by the job, including the LOG file and compilation listings. The default is 200 pages.	$JOB
/PRINT	SPRINT places the disk file created by the $CREATE card into the line printer output queue. If this switch is omitted, the file is not printed.	$CREATE
/PRIORITY:*nn*	Specifies the external priority of the job; the highest priority that can be specified is 62 (decimal). The default is 10.	$JOB
/RESTART	Specifies that the job may be restarted. /RESTART is the default.	$JOB

Switch	Meaning	Applicable Cards or Commands
/SEARCH	This switch specifies that the file is to be loaded in library search mode.	$INCLUDE
/SEQUENCE:*nn*	Specifies the job's sequence number within the input queue. The system creates a sequence number for the job if none is specified.	$JOB
/SUPPRESS	Trailing blanks are suppressed. The default is /NOSUPPRESS.	$COBOL, $CREATE, $DATA, $FORTRAN, $MACRO, $MODE, $TOPS20
/TIME:*hh:mm:ss*	The limit placed on the amount of CPU time used by the job. The default is 5 minutes.	$JOB
/UNIQUE:*n*	Specifies how jobs are to be protected from the effects of other jobs running in the same directory. n=0 means no protection; i.e., more than one Batch job may be run concurrently using the directory specified on the $JOB card. n-1 means only one Batch job at a time can be run using the specified directory. The default is 1.	$JOB
/WAIT	The job will wait for a response from the operator before resuming its processing.	$MESSAGE
/WIDTH:*nn*	Columns 1 through *nn* (inclusive) of each card are read. The remaining columns are truncated. If this switch is not specified or if no argument is specified, SPRINT will assume a value of 80. **NOTE** If no argument is specified, a warning message will be issued but SPRINT will still assume a value of 80 for the /WIDTH switch.	$COBOL, $CREATE, $DATA, $FORTRAN, $MACRO, $MODE, $TOPS20

APPENDIX B
MONITOR ERROR CODES

The error codes in Table B-1 are returned in AC on RUN and GETSEG monitor calls, in the right half of location E + 1 on 4-word arguments blocks of LOOKUP, ENTER, and RENAME monitor calls, and in the right half of location E + 3 on extended LOOKUP, ENTER, and RENAME monitor calls.

Table B-1
Monitor Error Codes

Code	Explanation
0	File not found, illegal filename (0,*), filenames do not match (UPDATE), or RENAME after a LOOKUP failed.
1	UFD does not exist on specified file structures. (Incorrect project-programmer number.)
2	Protection failure or directory full on DTA.
3	File being modified (ENTER, RENAME).
4	Already existing filename (RENAME) or different filename (ENTER after LOOKUP) or supersede (on a non-superseding ENTER).
5	Illegal sequence of monitor calls (RENAME with neither LOOKUP nor ENTER, or LOOKUP after ENTER).
6	1. Transmission, device, or data error (RUN, GETSEG only). 2. Hardware-detected device or data error detected while reading the UFD RIB or UFD data block. 3. Software-detected data inconsistency error detected while reading the UFD RIB or file RIB.
7	Not a saved file (RUN, GETSEG only).
10	Not enough memory (RUN, GETSEG only).
11	Device not available (RUN, GETSEG only).
12	No such device (RUN, GETSEG only).
13	Illegal monitor call (GETSEG only). No 2-register relocation capability.
14	No room on this file structure or quota exceeded (overdrawn quota not considered).
15	Write-lock error. Cannot write on file structure.
16	Not enough table space in free memory of monitor.

Table B-1 (Cont.)
Monitor Error Codes

Code	Explanation
17	Partial allocation only.
20	Block not free on allocated position.
21	Cannot supersede an existing directory (ENTER).
22	Cannot delete a non-empty directory (RENAME).
23	Sub-directory not found (some SFD in the specified path was not found).
24	Search list empty (LOOKUP or ENTER was performed on generic device DSK and the search list is empty).
25	Cannot create a SFD nested deeper than the maximum allowed level of nesting.
26	No file structure in the job's search list has both the no-create bit and the write-lock bit equal to zero and has the UFD or SFD specified by the default or explicit path (ENTER on generic device DSK only).
27	GETSEG from a locked low segment to a high segment which is not a dormant, active, or idle segment. (Segment not on the swapping space.)
30	Cannot update the file.
31	The low segment overlaps the high segment (GETSEG only).
32	Not logged in (RUN only).

APPENDIX C
CARD CODES

ASCII CARD CODES

Character	Octal Code	ASCII Card Punches	Character	Octal Code	ASCII Card Punches
NULL	00	12-0-9-8-1	'	47	8-5
CTRL-A	01	12-9-1	(	50	12-8-5
CTRL-B	02	12-9-2	)	51	11-8-5
CTRL-C	03	12-9-3	*	52	11-8-4
CTRL-D	04	9-7	+	53	12-8-6
CTRL-E	05	0-9-8-5	,	54	0-8-3
CTRL-F	06	0-9-8-6	-	55	11
CTRL-G	07	0-9-8-7	.	56	12-8-3
CTRL-H	10	11-9-6	/	57	0-1
TAB	11	12-9-5	0	60	0
LF	12	0-9-5	1	61	1
VT	13	12-9-8-3	2	62	2
FF	14	12-9-8-4	3	63	3
CR	15	12-9-8-5	4	64	4
CTRL-N	16	12-9-8-6	5	65	5
CTRL-O	17	12-9-8-7	6	66	6
CTRL-P	20	12-11-9-8-1	7	67	7
CTRL-Q	21	11-9-1	8	70	8
CTRL-R	22	11-9-2	9	71	9
CTRL-S	23	11-9-3	:	72	8-2
CTRL-T	24	9-8-4	;	73	11-8-6
CTRL-U	25	9-8-5	<	74	12-8-4
CTRL-V	26	9-2	=	75	8-6
CTRL-W	27	0-9-6	>	76	0-8-6
CTRL-X	30	11-9-8	?	77	0-8-7
CTRL-Y	31	11-9-8-1		100	8-4
CTRL-Z	32	9-8-7	A	101	12-1
ESCAPE	33	0-9-7	B	102	12-2
CTRL-\	34	11-9-8-4	C	103	12-3
CTRL-]	35	11-9-8-5	D	104	12-4
CTRL-^	36	11-9-8-6	E	105	12-5
CTRL-¯	37	11-9-8-7	F	106	12-6
			G	107	12-7
SPACE	40		H	110	12-8
!	41	12-8-7	I	111	12-9
‘	42	8-7	J	112	11-1
#	43	8-3	K	113	11-2
$	44	11-8-3	L	114	11-3
%	45	0-8-4	M	115	11-4
&	46	12	N	116	11-5

Character	Octal Code	ASCII Card Punches	Character	Octal Code	ASCII Card Punches
O	117	11-6	g	147	12-0-7
P	120	11-7	h	150	12-0-8
Q	121	11-8	i	151	12-0-9
R	122	11-9	j	152	12-11-1
S	123	0-2	k	153	12-11-2
T	124	0-3	l	154	12-11-3
U	125	0-4	m	155	12-11-4
V	126	0-5	n	156	12-11-5
W	127	0-6	o	157	12-11-6
X	130	0-7	p	160	12-11-7
Y	131	0-8	q	161	12-11-8
Z	132	0-9	r	162	12-11-9
[	133	12-8-2	s	163	11-0-2
\	134	0-8-2	t	164	11-0-3
]	135	11-8-2	u	165	11-0-4
^	136	11-8-7	v	166	11-0-5
–	137	0-8-5	w	167	11-0-6
,	140	8-1	x	170	11-0-7
a	141	12-0-1	y	171	11-0-8
b	142	12-0-2	z	172	11-0-9
c	143	12-0-3	{	173	12-0
d	144	12-0-4	|	174	12-11
e	145	12-0-5	}	175	11-0
f	146	12-0-6	~	176	11-0-1
			DEL	177	12-9-7

INDEX

INDEX (Cont.)

INDEX (Cont.)

DECsystem-20 Batch
Reference Manual
DEC-20-OBRMA-A-D

READER'S COMMENTS

NOTE: This form is for document comments only. Problems with software should be reported on a Software Problem Report (SPR) form.

Did you find errors in this manual? If so, specify by page.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Is there sufficient documentation on associated system programs required for use of the software described in this manual? If not, what material is missing and where should it be placed?

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Non-programmer interested in computer concepts and capabilities

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code or Country _____

If you require a written reply, please check here. ☐

Please cut along this line.

---------- Fold Here ----------

---------- Do Not Tear - Fold Here and Staple ----------

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

digital

Software Communications
P.O. Box F
Maynard, Massachusetts 01754

DECsystem-20 Batch
Reference Manual
DEC-20-OBRMA-A-D

READER'S COMMENTS

NOTE: This form is for document comments only. Problems with software should be reported on a Software Problem Report (SPR) form.

Did you find errors in this manual? If so, specify by page.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Is there sufficient documentation on associated system programs required for use of the software described in this manual? If not, what material is missing and where should it be placed?

Please cut along this line.

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Non-programmer interested in computer concepts and capabilities

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code or Country _____

If you require a written reply, please check here. ☐

---------- Fold Here ----------

---------- Do Not Tear - Fold Here and Staple ----------

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

digital

Software Communications
P.O. Box F
Maynard, Massachusetts 01754


digital